

Objectives:

- Learn how to use (and practice) Tree ADT (binary and/or binary search tree)
- Practice using Recursion to traverse Tree Objects.
- Practice refactoring operations (replacing a data structure with another)
- Working with projects with multiple files.
- Program comprehensions, impact analysis, debugging, and refactoring
- Practice using exceptions and exception handling in C++ when applicable.
- Practice using Iterators with nonlinear data structures.

Problem Description (Optional) [30 Points]

In this project, you are expected to rewrite your system from assignment 6 (or 5) using either Binary Tree or Binary Tree ADT. The implementations are already posted on Canvas and have been already discussed and explained in class. I recommend using the Binary Search ADT or the Vector-Based Binary tree. The full code is available on Canvas. However, feel free to enhance the implementation when fit.

Again, this is supposed to be a subsystem from car rental software. The system is expected to help the inventory automate its vehicle reservation and query processes (without getting involved in the customer information). It uses a data structure to keep track of the vehicle inventory in the system. Moreover, vehicle information is already provided in a text file, and no need to worry about inserting new vehicles into the file. However, the system needs to update the file after each reservation or return process. The good news is that most of the work is done on your behalf.

The given program performs at least the following operations:

1. Print All of the registered vehicles. (available and already reserved)
2. Print list of the available vehicles.
3. List of the reserved vehicles (rented currently by customers).
4. Check availability.
5. Reserve a vehicle.
6. return a reserved vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit

For simplicity, only two major components (classes) were used to develop the system: **Vehicle** and **Inventory** with a list of vehicles stored in a file.

An application file that contains the main function is provided as well. The main function controls the selection menu and reads the inventory information from the file (vehicles.txt).

Data structures expected to be used in this project: Either Binary Search Tree or Vector-Based Binary tree that are posted on Canvas.

Your job:

Part 1: Refactoring: Replace the underlining NodeList in classes Vehicle and Inventory with objects of Binary Search Tree (or Vector-Based Binary Tree).

For the Vehicle class, a binary search tree is used for storing the extra feature each vehicle has. You can use the ID and a key for sorting the vehicles. Of course, you need to update all of the affected methods by this replacement.

The same thing for the class Inventory. a binary search tree is used for storing the vehicles' information read from the file. Of course, you need to update all of the affected methods by this replacement (e.g., insertion methods, searching methods).

.


Part 2: Refactoring: reimplementing most of the methods so that they work with the binary search tree (or vector-based binary tree).

examples:

- void Inventory::printList()
- void Inventory::printResevedList()
- void Inventory::printAvailableList()
- bool Inventory::found(int seatsNo)

Of course, many traversal algorithms (in-order, post-order, and pre-order) have been already implemented for you and posted on Canvas. You just need to use them to implement your methods.

Supporting Code Files

- You can use your code from assignment 6 (or 5)
- Or you can use this zipped incomplete project is provided: [Project6_OldVersion.zip](#) 
- [Download Project6_OldVersion.zip](#)
- Replace the dynamic arrays with Tree objects and update the methods to work with the tree objects.

Clearing the console screen can be done using: `system("cls");`

Example Screens:

This is how records are printed:

```
-----  
ID: 10004  
Make: Dodge  
Model: Grand Caravan  
Vumber of seats: 7  
Availability: available  
Extra Features[3]:  
[AC, GPS, DVD]  
-----  
-----
```

```
ID: 10006  
Make: Chevrolet  
Model: Equinox  
Vumber of seats: 5  
Availability: available  
Extra Features[3]:  
[AC, DVD, Back Camera]  
-----  
-----
```

[Main Menue:](#)

```
0. Clear Screen..
1. Print All of the registered vehicles.
2. Print list of the available vehicles.
3. List of the reserved vehicles.
4. Check availability.
5. Reserve a vehicle.
6. return a vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit.
Your Selection -->
```

Check Availability:

```
0. Clear Screen..
1. Print All of the registered vehicles.
2. Print list of the available vehicles.
3. List of the reserved vehicles.
4. Check availability.
5. Reserve a vehicle.
6. return a vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit.
Your Selection --> 4
Please Enter number of seats? 5_
```

Reserve A Vehicle:

C:\U5

```
0. Clear Screen..
1. Print All of the registered vehicles.
2. Print list of the available vehicles.
3. List of the reserved vehicles.
4. Check availability.
5. Reserve a vehicle.
6. return a vehicle.
7. Print All of the registered vehicles to a text file.
8. Exit.
Your Selection --> 5
Please Enter the Vehicle ID: █
```

Useful Videos: Recordings:

Tree ADT:

Binary Search Tree:

[Watch 'CS 244-002, Data Structures Virtual Class Meeting.' | Microsoft Stream \(https://web.microsoftstream.com/video/2166a32b-9f70-489f-bc39-2f49441350fe \) \(Links to an external site.\)](https://web.microsoftstream.com/video/2166a32b-9f70-489f-bc39-2f49441350fe)

[Watch 'CS 244-002, Data Structures Virtual Class Meeting.' | Microsoft Stream](#)